

MPEG RVC AVC Baseline Encoder Based on a Novel Iterative Methodology

Hussein Aman-Allah, Ehab Hanna, Karim Maarouf, Ihab Amer
Laboratory of Microelectronic Systems (GR-LSM), EPFL
CH-1015 Lausanne, Switzerland
{hussein.aman-allah, ehab.hanna, karim.maarouf, ihab.amer}@epfl.ch

Abstract

With the emergence of new generations of multicore architectures, the need for efficient multimedia algorithm implementations has become critical. This paper describes a new methodology for efficient implementations of algorithms targeting reconfigurable architectures. The Reconfigurable Video Coding (RVC) standard aims to provide a framework allowing a dynamic development, implementation and adoption of standardized video coding solutions with features of higher flexibility and reusability. RVC-CAL actor language is a dataflow language that makes better use of the multicore and parallel architectures. The proposed design flow methodology follows an iteration-based implementation model rather than the traditional sequential model. Analysis, design, development, simulation, testing and adaptation are performed with every iteration ending up with a functional “version” of the algorithm. A case study is conducted to illustrate the productivity of the proposed methodology in which the implementation of an AVC baseline encoder on a Xilinx Virtex 5 XC5VLX50T FPGA demonstrated for intra prediction architecture search space co-exploration.

1. Introduction

The increasing complexity of video coding standards has led to the development of the platforms corresponding to these standards and their evolution into multicore and multiple component parallel architectures. Nevertheless, algorithms are still being specified in the same monolithic sequential models, mostly in C/C++ that are unable to exploit the full capabilities featured by the advances in the target architectures. Thus, the MPEG RVC standard while addressing the reusability and reconfigurability of the different MPEG standards, defines RVC-CAL to be the normative language of its video tool library (VTL). The nature of RVC-CAL as a dataflow language

allows it to exploit the parallelism offered by the new platforms while at the same time allowing for the reusability and reconfigurability of the FUs provided in the RVC VTL.

Simulating parallelism is one of the main advantages of using dataflow programming over sequential programming, making it more suitable for implementations targeting multicore architectures [1]. Unlike C/C++ implementation where parallelism has to be explicitly defined; the developer has more time to focus on algorithm-specific details and architecture-related optimizations. Nevertheless, the inconsistency arising from starting with an RVC-CAL description and following the sequential implementation methodology often leads to unpredictable results.

This paper presents an iterative *design flow* methodology for the implementation of efficient reconfigurable architectures as opposed to sequential models that do not suit parallel architectures, supported by a case study of an MPEG RVC AVC baseline encoder [2].

The paper is structured as follows: Section II presents the proposed methodology. In section III, the case study covering AVC inter prediction, intra prediction and entropy coding modules is examined to demonstrate this methodology. The modeling and synthesis results are presented and analyzed in section IV. Finally, section V concludes the paper.

2. An iterative methodology

The proposed methodology recommends a *design flow*, along with *tools* and *practices* that when applied precisely on a given algorithm, results into an optimized implementation of the target architecture.

Given an algorithm to be implemented, first the appropriate FUs are selected from the VTL. An RVC-CAL description is then provided to assemble the FUs and reconfigure them by adding new functionalities that are not yet provided by the VTL. Test cases are

fed to the RVC-CAL description to verify it against the standardized reference SW.

Once the RVC-CAL description is validated, the CAL2HDL tool performs resource scheduling and optimizations that result in an efficient HDL description [3]0. A test bench is provided to verify that the generated HDL functionality sustains the one provided by the RVC-CAL description in the previous step. In case of any anomalies, a revision of the RVC-CAL description is performed to make sure that the definitions are unambiguous for the CAL2HDL tool. Optimizations are possibly applied to the RVC-CAL description to allow for the generation of HDL code of better exploitation for the available hardware resources, based on observations of the previously generated HDL code.

The next step is to synthesize the generated HDL code into hardware implementation targeting a specific architecture. The output is simulated and examined. If after the mapping, and routing performed by the synthesis tool the implementation does not meet the target architectural specifications, in terms of resource consumption, throughput, frequency or other parameters, a revision of the RVC-CAL description is performed for optimizations. When the synthesized implementation meets its target architecture design goals, the process terminates and the implementation is ready.

The main advantage provided from the proposed design flow and the associated tools is that several iterations of the steps described in Figure 1 can be performed in considerably less time than the traditional sequential models for several reasons.

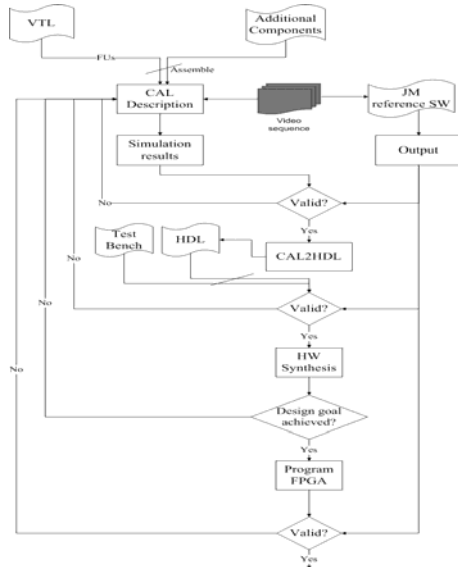


Figure 1 - The proposed design flow

First, the bottlenecks and anomalies are detected early in the process, because of RVC-CAL's ability to simulate parallel and multicore architectures. Second, the implementation passes through several optimizations on different levels. Third, the modifications are always performed on the RVC-CAL level. Thus, it needs less time and frequent architectural adaptations are feasible because of the strong abstraction and encapsulation properties exhibited by CAL. The proposed methodology allows for exploration of the resources/throughput search space. A set of transitions are explored until the implementation meets the algorithm/architecture design goals [4].

The initial RVC-CAL description is usually a direct implementation of the algorithm that aims at creating a starting point for simulation. The synthesis result may not meet the target specifications, since no optimization iterations have been performed. Several optimization cycles are performed which moves the implementation closer and closer to the target specifications.

3. RVC AVC baseline encoder case study

The proposed methodology along with the tools and practices are applied to a case study of an MPEG RVC AVC baseline profile encoder. The aim of the case study is to demonstrate the contribution of the proposed approach on the efficiency of the implementation.

3.1. Inter prediction

The AVC Inter prediction module is used to estimate the motion occurring between frames. This module aims at removing the temporal redundancy that exists between frames in order to achieve a high compression of the video sequence [5]. Figure 2 shows the inter prediction RVC-CAL Network. The two main steps in inter prediction are motion estimation and motion compensation. The motion estimation calculates the motion between frames via full search macro-block matching and returns a motion vector describing the displacement between the current block and the best matching reference block. The motion compensation uses the current block and the best matching reference block to calculate the residual error. The residual error is then passed to the transform coding module to be compressed, reconstructed and finally sent back to the motion compensation module to reconstruct the current

The diagram illustrates the proposed video coding architecture. It starts with a 'Video sequence' (represented by a stack of blue rectangles) which is fed into a 'Video Reader'. The 'Video Reader' outputs '1 Frames' to an 'Intra-prediction Module' and 'P Frames' to a 'Memory Module'. The 'Intra-prediction Module' outputs 'Reconstructed Reference Frames' to the 'Memory Module'. The 'Memory Module' outputs 'Reconstructed current Block' to a 'Motion Compensation' block. The 'Motion Compensation' block also receives 'Motion Vectors' from a 'Motion Estimation' block and outputs 'E' to a 'Transform Coding Module'. The 'Transform Coding Module' outputs 'RE' to the 'Motion Estimation' block. The 'Motion Estimation' block also receives 'Motion Vectors' from a 'Motion Vector Predictor' block. The 'Motion Vector Predictor' block receives 'Differential Motion vectors' from the 'Memory Module' and outputs 'Motion Vectors' to the 'Motion Estimation' block. The 'Motion Estimation' block outputs 'Search window and Current Block' to the 'Memory Module'. The 'Memory Module' also receives 'Current Block and Best Matching Reference Block' from the 'Motion Estimation' block. The 'Motion Compensation' block outputs 'Motion Vectors' to the 'Motion Estimation' block. The 'Motion Estimation' block outputs 'Motion Vectors' to the 'Motion Compensation' block.

As shown in Figure 2, the video reader reads individual frames from a video sequence and either passes them directly to the memory as P-frames or to the intra prediction module, which first encodes and reconstructs the I-frame before storing it in the memory. Finally, inter prediction calculates the differential motion vectors based on the difference between the actual and predicted motion vectors [6]. The modules are moderated by the inter prediction control unit (IPCU).

In intra prediction, predictors are formed from previously encoded and reconstructed Macro Blocks (MBs). In AVC luma prediction, predictors can be produced either for every 4x4 block in a MB (9 modes) or for the MB as a whole (4 modes). The encoder selects the mode which minimizes the difference between the predictor and the block to be encoded [6].

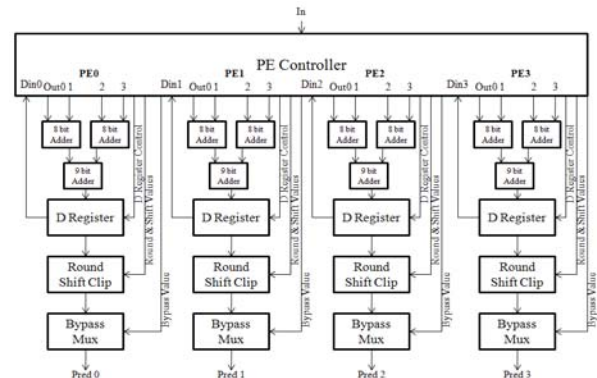


Figure 3 [7] shows the RVC-CAL network for an intra prediction module.

The design is based on [7], where reconfigurable processing elements (PEs) are used to generate predictors for all the modes instead of one PE for each mode. There are four PEs, which generate four predictors in parallel. Using this four-parallel architecture achieves a good tradeoff between throughput and resource usage. Each PE consists of a series of Adders, Registers and Shifters which calculate the value of the predictor. The inputs to the PEs are determined by the main actor in the network: the *PE Controller*. It selects the suitable inputs depending on the prediction mode and the position of the sample to be predicted. A more detailed explanation of this module can found in [8].

The AVC baseline profile uses Exp-Golomb to encode all syntax elements except for the residual data that are encoded using Context Adaptive Variable Length Coding (CAVLC). Exp-Golomb codes are variable-length binary codes that are constructed systematically as thoroughly explained in [9]. Figure 4 [2] shows the RVC-CAL network implementing the Exp-Golomb.

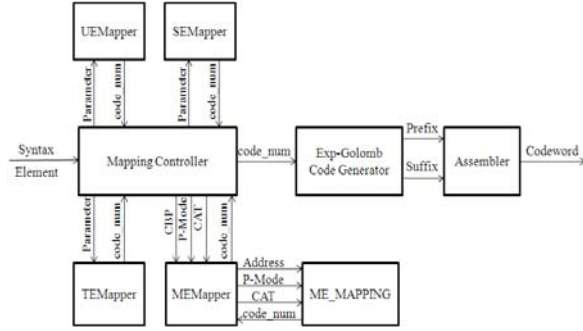


Figure 4 - Exp-Golomb RVC-CAL network

The Exp-Golomb module shown in Figure 4 is implemented in RVC-CAL as a simple network with one input port, which receives the parameter token to be encoded and one output port which outputs the codeword serially. The Exp-Golomb network connects nine different actors, out of which four actors perform the tasks of the four different mapping modes. A *controller* moderates the mapping actors and performs the mapping mode decision. The different mappings can be executed in parallel since a different actor is responsible for each. Thus, with pipelining at the controller, and parallel execution at the mapping actors, the throughput improves significantly.

Similarly, the RVC-CAL CAVLC model shown in Figure 5 [2] is implemented with slightly more actors. Since CAVLC switches different VLC look up tables (LUTs) to encode the different syntax elements based on values of previously coded elements. Each LUT exploits the statistical properties of the syntax elements it is designed to encode. Accessing the LUTs can be sometimes performed in parallel and the different actors execute and prepare their output independent of the sequential flow of the algorithm. The Assembler actor is then responsible for compiling the output tokens from the different actors and producing the encoded stream serially.

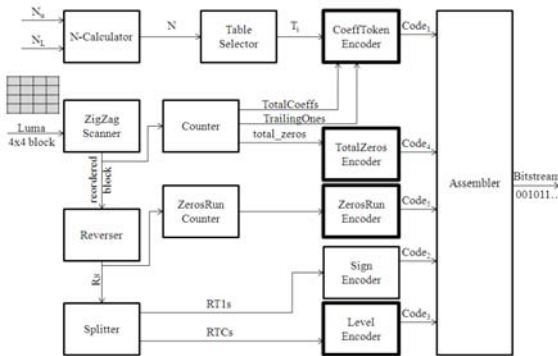


Figure 5 - CAVLC RVC-CAL network

4. Results and analysis

The results of applying the proposed methodology to the RVC encoder can be presented on two levels; the modeling level including the development efforts consumed and the productivity gained, and the implementation level in which the RVC-CAL implementation is synthesized and implemented on an FPGA.

4.1. Modeling results

One of the major RVC advantages is that it accompanies its normative description language (RVC-CAL) with many supporting tools that enable automatic code generation into software (CAL2C) and hardware (CAL2HDL) [8]. The results section in [2] shows the lines of code, development time and number of developers consumed in the RVC-CAL, C and VHDL implementations correspondingly. Although the number of iterations performed on the RVC-CAL implementation exceed the other two implementations with a factor of four or five, the development effort consumed is much less. Such saving reflected in Figure 6 contributes directly to production costs and Time to Market (TTM).

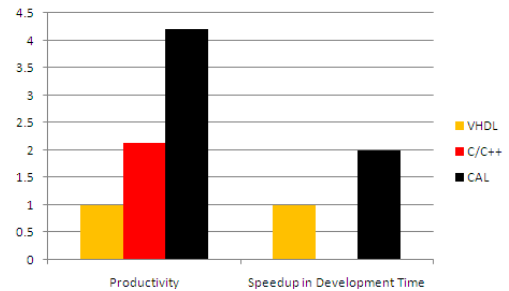


Figure 6 - Gain from using the RVC-CAL implementation

Figure 6 shows a comparison, normalized with respect to the VHDL implementation. The RVC-CAL implementation is as double as productive as the C/C++ sequential model and four times as productive as the VHDL behavioral implementation. Additional enhancements to RVC-CAL will embed built-in functions that would even double the RVC-CAL productivity.

4.2. Search space co-exploration

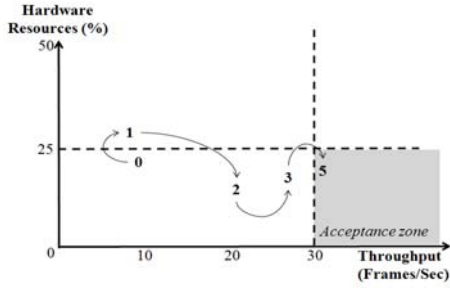


Figure 7 - Intra prediction search space co-exploration

Following the proposed methodology, the intra prediction module initial design was validated against the AVC reference software. However, upon synthesis, the resulting FPGA implementation did not achieve throughput requirements for real-time SDTV (720x480). Therefore, optimization iterations were needed. Figure 7 and Table 1 show the resource/throughput states at the end of the optimization iterations. Optimization proceeds by searching for the critical path and splitting it. This could be done by dividing actors or actions. In the first iteration, the *PE Controller* was split into two actors and their outputs multiplexed to the PEs. However, this multiplexing increased the critical path which made the throughput worse. Therefore, in the second iteration cycle, instead of multiplexing, the PEs were duplicated. In the third and fourth iterations, the *PE Controller* was split into three and four actors. Splitting into four actors resulted in an implementation that exceeded the FPGA resources. However, having three *PE Controllers* resulted in the best tradeoff between throughput and resources. Therefore, the fifth iteration targeted the actions in each of the three *PE Controllers*. There is a tradeoff between the number of actions and size of the state machine. A large number of small actions complicate the state machine, which increases the control logic for it. On the other hand, a small number of actions simplify the state machine, but the control logic in the actions themselves is complicated. This means that the right balance between action and state machine size must be found for each actor. Therefore, in the fifth iteration, the simple actions were grouped and the complicated actions were divided to achieve a balance between state machine size and action size. This led to meeting both throughput and resource requirements.

4.3. Synthesis results

The gain achieved on the modeling level in terms of lines of code is echoed as well on the hardware implementation level. The HDL model is generated from the presented RVC-CAL model using the CAL2HDL tool. The HDL code is synthesized using Xilinx ISE targeting a Xilinx Virtex 5 FPGA.

Table 1 shows an example of the proposed optimization iterations applied to the intra prediction module showing synthesis results at the end of each iteration. The methodology performs several optimization iterations until the target design goals are achieved.

Table 1 - Intra prediction exploratory iterations

Iteration	CLK Frequency (MHz)	# of Registers	# of LUTs	Throughput (Frames/s)
0	49.615	5533	7761	10
1	46.755	6892	9021	9
2	85.69	4554	5079	21
3	110.011	5755	4586	27
4	Exceeded Available Resources			
5	120.221	5477	6293	30

5. Conclusion

In this paper, a methodology for efficient implementation of algorithms targeting reconfigurable architectures was presented. The methodology defines an iterative design flow, a set of tools and practices for architecture exploration and optimization. The use of RVC-CAL as the modeling language for the proposed methodology, exploits parallelism and reconfigurability supported by the newer generations of multimedia platforms. In addition, the strong abstraction and encapsulation properties exhibited by RVC-CAL allow for faster optimization iterations.

An MPEG RVC AVC baseline encoder case study was provided to demonstrate the gain achieved from using the proposed methodology, on both the modeling and implementation levels. The results show that the proposed approach minimizes the time and effort consumed during development. The results are also reflected on the hardware implementation level, taking Intra prediction as example.

6. References

- [1] Bhattacharyya, S., Brebner, G., Janneck, J., Eker, J., von Platen, C., Mattavelli, M., Raulet, M. OpenDF – A Dataflow Toolset for Reconfigurable Hardware and Multicore Systems. First Swedish Workshop on Multi-Core Computing, 2008.

- [2] Aman-Allah, H., Hanna, E., Maarouf, K., Amer, I. Towards a Comprehensive RVC VTL: A CAL Description of an Efficient AVC Baseline Encoder. IEEE International Conference on Image Processing (ICIP 2009), 2009.
- [3] Janneck, J., Miller, I. D., Parlour, D. B., Roquier, G., Wipliez, M., Raulet, M. Synthesizing hardware from dataflow programs: An MPEG-4 simple profile decoder case study. IEEE Workshop on Signal Processing Systems (SiPS 2008), 2008.
- [4] Lucarz, C., Faure, P., Roquier, G., Mattavelli, M., Noel, V., Amer, I., and Alisafee, M. "ACTORS" European Project, technical report, 2009-01, Micro-Electronic Systems Laboratory, Ecole Polytechnique Federale de Lausanne (EPFL), 2009.
- [5] Yang, W. (2003). An Efficient Motion Estimation Method for MPEG-4 Video Encoder. IEEE Transactions on Consumer Electronics, 49(2).
- [6] Richardson, I. E. G. H.264 and MPEG-4 Video Compression. Aberdeen. Wiley, 2003.
- [7] Huang, Y., Hsieh, B., Tung-Chien C., Chen, L. Analysis, Fast Algorithm, and VLSI Architecture Design for H.264/AVC Intra Frame Coder. IEEE Transactions on Circuits and systems for Video Technology, 15(3), 2005, 378-401.
- [8] Lucarz, C., Mattavelli, M., Wipliez, M., Roquier, G., Raulet, M., Janneck, J., et al. Dataflow/Actor-Oriented language for the design of complex signal processing systems. (DASIP, 2008)
- [9] Silva, T., Vortmann, J., Agostini, L., Bampi, S., Susin, A. FPGA Based Design of CAVLC and Exp-Golomb Coders for H.264/AVC Baseline Entropy Coding. 3rd Southern Conference on Programmable Logic (SPL07), 2007.